MUSE

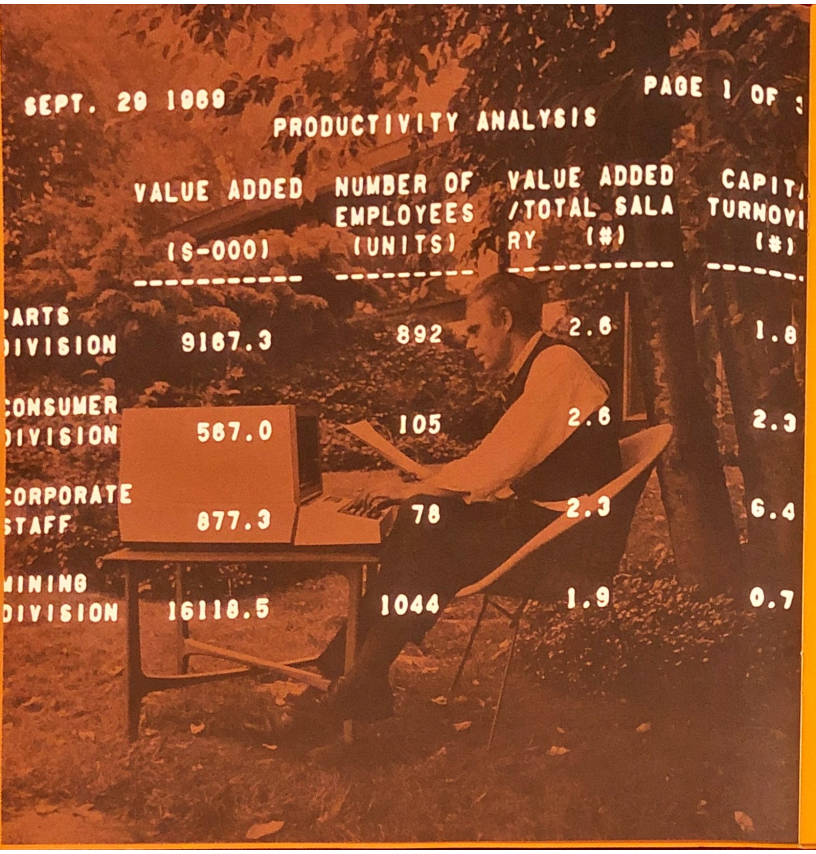
EXECUTIVE SUMMARY

Copyright © 1970 by Meta-Language Products, Inc. All rights reserved. No part of this publication may be reproduced by any means, nor transmitted, nor translated into a computer language without the written permission of Meta-Language Products, Inc.

META-LANGUAGE PRODUCTS, INC.
IS PROUD TO ANNOUNCE MUSE

an interactive information system for solving non-routine management problems. MUSE, an acronym for *M*achine *U*ser *S*ymbiotic *E*nvironment, is a trademark of Meta-Language Products, Inc.

MUSE was developed because there existed a need for a generalized management information system utilizing: time-shared computers, natural language interaction, efficient data management and automatic report-generation. No other management information system has satisfactorily provided all of these features.


SEPT. 20 1969
PAGE 1 OF
PRODUCTIVITY ANALYSIS
VALUE ADDED ($-000)
NUMBER OF EMPLOYEES (UNITS)
VALUE ADDED /TOTAL SALARY (#)
CAPITA TURNOVI (#)
PARTS DIVISION 9167.3 892 2.6 1.8
CONSUMER DIVISION 567.0 105 2.6 2.3
CORPORATE STAFF 877.3 78 2.3 6.4
MINING DIVISION 16118.5 1044 1.9 0.7

With MUSE, each of many executives in the same company is able to sit at a console ... and simultaneously communicate directly and creatively —actually interact—with the computer...

BRIEFLY STATED

MUSE is a new, unique and "natural" computer language *in English*, created expressly for use by management in an interactive environment.

MUSE consists of ordinary English words and grammar; therefore, it does not require the extensive special training necessitated by most computer languages such as FORTRAN and COBOL. With MUSE, each of many executives in the same company is able to sit at a console in his private office and simultaneously communicate directly and creatively—actually interact—with the computer in a time-sharing environment.

MUSE, however, is *more* than a "natural" language permitting interaction between the user and the computer. It is a complete proprietary management information system including:

A simple data-loading and updating procedure
While management should decide what kind of information is needed, the executive-user does *not* have to be concerned about actually loading or maintaining the data base.

An automatic report-generator
The specific data that executives need are produced immediately in the form of reports, graphs or simple answers.

Computer programing, which is ordinarily difficult, complicated, and dependent upon the services of highly skilled technicians, is "built into" MUSE.

The most vital aspects of corporate information—those on which important non-routine decisions are based—are immediately accessible to management. This is an expansion of computer capability far beyond the handling of routine problems such as billing, invoicing, payroll, and accounts receivable.

It is also a significant extension of current decision-making applications which even those companies sophisticated in the use of the computer must handle in a rigid, structured way.

In fact, it is anticipated that sophisticated companies will be the most receptive to MUSE. These companies are already keenly aware of the benefits that can accrue from a *total* information system: better, faster, more accurate decisions; improved control; options to decentralize and thereby create multiple profit centers; improved communications, etc.

UNTIL NOW THE TOTAL INFORMATION SYSTEM HAS BEEN JUST A CONCEPT.

"MUSE" MAKES IT A REALITY.

Meta-Language Products, Inc., offers MUSE on a licensing basis that is unique in many respects:

- To insure that MUSE will be used to the greatest advantage, a *consultant* will be stationed in the customer's offices for six months. The consultant will aid in the development of the corporate data base and applications of MUSE. He will also conduct seminars for the personnel actually using the system.
- Systems maintenance support will be provided during the full term of the license and free system updates will be provided until February 1, 1972.
- The company will stand behind its product to the extent of offering a *money-back guarantee* if the MUSE system does not perform according to a supplied set of specifications.

THE NEED FOR MUSE

For some time, there has been a growing realization that top management is not fully utilizing the capabilities of third generation computers for the handling of non-routine problems.

Certainly computers are being put to good use in high-volume, repetitive, routine data-processing.

And some companies are taking this high-volume "batch" processing a step further by having the computer report "exceptions" to operational plans.

But this presents management with additional problems because the system churns out mountains of detailed information. This ends up in stacks of reports, the sheer volume of which makes them ineffectual as a basis for action.

Even if the manager had the time to act on an "exception," the consequent decision would be difficult and possibly dangerous.

Assume that the manager is in the marketing area. Before he can take intelligent action he must know whether "exceptions" to plan are the result of deviations in sales effort, consumer buying power, advertising or promotional expenditures, competitive activities or other factors.

The interactive environment made possible by time-sharing was heralded as providing the answer to such non-routine problems. But, prior to MUSE, the development of time-sharing hardware was far beyond that of software.

Now, with the use of MUSE in conjunction with time-sharing, the marketing manager in the above example can diagnose his problems by (1) marshaling only relevant data for the inspection of alternatives, (2) reducing the bulk of the data by, for example, screening, ranking, cross-referencing or correlating and (3) calling for a report that will pin-point the problem and enable him to make an intelligent decision.

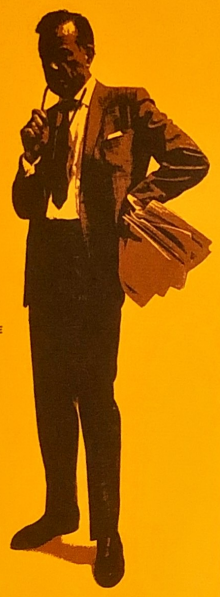
PRIOR TO MUSE

According to James R. Bradburn, RCA Executive Vice President, Information Systems (June, 1969), "If it is to make a substantive difference in our lives, the computer must become accessible to people at all levels of technical proficiency, not just professionals or programers."

In recognition of this need, one or two programing languages—such as BASIC—were developed as a good first step. Nevertheless, these languages show persistent drawbacks:

- Non-English use of symbols, rigid phrase structures, and unnatural grammatical constructions predominate.
- Each program can refer to only a small amount of data. If a particular piece of data is needed by more than one user it must be duplicated.
- Data are not referred to by name but by location within files—causing the user a large cross-referencing and indexing task.
- Report-generating requires a certain amount of setting-up of the output form, which is a tedious job. Moreover, the user must supply the logical flow of events to create a report.

These drawbacks have continued to be prevalent in other later developments both in languages and management information systems, some of which are called "file management systems." (As the term suggests, these systems are severely limited.) In addition, other systems have been dependent upon highly complicated and technical data-loading and maintenance procedures; they have static data libraries; they



"...the computer must become accessible to people at all levels of technical proficiency, not just professionals or programers."

The unique linguistic simplicity and statistical power of MUSE make it highly applicable for problem-solving in many corporate areas...

AN EASYGOING YET POWERFUL SYSTEM

are not interactive and they have been unable to recognize units and scaling. Finally, the implementation of some of these systems on the computer requires a large amount of expensive storage.

The user of MUSE, however,

- communicates directly with the computer in conversational English,
- refers to data by name,
- draws on an extensive and dynamic corporate data base (can be over a billion characters of data), which will have been loaded by a simple mechanism,
- and receives a clear and easily understood report automatically, based on the data and the man-computer dialogue, within a matter of seconds.

With MUSE, the computer is not only an extension of man—it is a participant in a printed dialogue. Similar to the way two people communicate by voice on the telephone, the man and the computer with MUSE communicate in print. The executive using MUSE can give commands, make statements, or ask questions; the computer, as if it were another person, responds appropriately or asks other questions.

The executive can use MUSE immediately. He can ask the computer, "HOW DO I USE MUSE?" and the computer will tell him. He can ask what to do next, and the computer will give instructions. This is a completely direct interactive environment, utilizing an easy and highly adaptable language.

MUSE is easy to use, however, *only because of the enormous complexity of the system.* MUSE shifts the burden of communication from the user to the system itself. It is a flexible language, eminently suited to the handling of non-routine problems and decisions. The unique linguistic simplicity and statistical power of MUSE make it highly applicable for problem-solving in many corporate areas: banking, insurance, manufacturing, retailing, transportation, utilities; also for government agencies, universities, unions, foundations and other non-profit organizations.

Within such varied corporate and organizational structures, MUSE is extremely effective in finance, marketing, administration, distribution, production, personnel, corporate planning and other types of analyses and forecasting.

Various statistical capabilities such as correlation, ratio analysis, indexing, seasonal adjustment and exponential smoothing are easily accomplished and, conversely, MUSE can also handle simulation model building.

The preceding summary of MUSE has scarcely scratched the surface. (Even as this is being written, the capabilities of MUSE are being rapidly expanded.) Some of the questions about MUSE that users want answered appear on the following pages. Meta-Language Products suggests that further detailing be accommodated at presentations. These can be arranged upon request through:

Harold C. Kurfehs, Meta-Language Products, Inc.,
4 East 43rd Street, New York, N.Y. 10017.

The most important thing to the user of MUSE is:

"THE *COMPUTER* IS NO LONGER THE PROBLEM.
THE *PROBLEM* IS THE PROBLEM."

QUESTION How is MUSE different from other computer software products?

ANSWER MUSE is different from other software in that it incorporates a fundamental language which will be implemented on a *wide variety* of time-shared computers made by various manufacturers. *It will not be dependent upon one particular computer or one particular manufacturer.*

Also, MUSE is not simply a program or series of programs for particular applications. Programing facility is contained within the language, thus extending the range of its application far beyond any one type of problem. Every user-computer dialogue in MUSE, whether long or short, simple or complex, is an applications program.

MUSE is also the only system providing (1) a simple mechanism for data-loading and updating which can be handled by a junior programer without writing conversion routines, (2) automatic report formating with no user-intervention necessary, (3) great flexibility of synonym and phrase structure within the language itself, (4) simultaneous unit and scaling calculations and (5) a dictionary with definitions of every word used in the system. The words themselves enable executives to determine quickly what data are currently available.

In addition, MUSE will take up a relatively small amount of storage since it is written in the assembly language of the computer model on which it is used. The result is that MUSE is much faster and much less space-consuming than systems written in algorithmic languages like JOVIAL and LISP.

MUSE provides simultaneous access by many users to an extensive *private* corporate data base. To complement the private data base the user will also have access to *public* data through at least one major independent time-sharing service.

As previously stated, Meta-Language Products, Inc., is unique in providing a consultant for six months and in offering a money-back guarantee.

QUESTION Just what is meant by a "natural" or "plain" computer language?

ANSWER MUSE is a "natural" computer language because it uses natural English. The words are in English; the grammar is that of English. When the executive and the computer "talk" MUSE, they talk English, using three main sentence forms: Commands, Questions and Descriptions.

Questions and Descriptions handle the job of information retrieval and computational work. Commands supply the user with control over the direction of the dialogue.

A Command, which ends in an exclamation point, is used to record or act upon a dialogue; create or modify synonyms and phrases; request or change definitions; input data; and modify, list or delete parts of a dialogue. The following are examples of Commands:

PRINT THE DEFINITION OF "CASH FLOW"!

EQUATE "SOM" WITH "SHARE OF MARKET"!

PLEASE PRINT THIS DIALOGUE!

REPLACE THE FIFTH WORD IN THE PREVIOUS SENTENCE WITH "ROUND"!

CHANGE THE VALUE OF "PRODUCTION FOR THE INDIAN RIVER GROVE FOR THE FOURTH QUARTER OF 1967"!

A Question in MUSE, which ends in a question mark, can be free-form, computational or referential. Answers to free-form questions (how and why) explain how one uses MUSE. Questions can also range from the very simple to the very complex. Responses are immediate. Examples of Questions are:

Free-form:

WHY DID YOU GIVE ME THAT MESSAGE?

HOW IS A REPORT GENERATED?

HOW DO I GET A DEFINITION FROM THE DICTIONARY?

HOW DO I CORRECT A PIECE OF DATA?

Computational:

WHAT WILL THE ESTIMATED NET WORTH OF IBM BE FOR 1970?

WHAT IS THE ESTIMATED COST FOR A POINT BARROW REFINERY IF THE CAPACITY EQUALS 5000 BBL PER DAY AND THE CONSTRUCTION DATE EQUALS THIS YEAR?

WHAT IS THE SEASONAL VARIATION OF THE PRODUCTION IN THE DURABLE GOODS SECTOR FOR NOVEMBER OF LAST YEAR?

WERE SALES FOR ANY SUBSIDIARY COMPANY FOR 1967 GREATER THAN $95 MILLION OR 5% OF THE PARENT?

Referencing:

WHAT IS THE INVENTORY LEVEL FOR PRODUCT 68?

WHAT WAS THE VALUE ADDED FOR THE PARTS DIVISION FOR 1968?

WHAT ARE ACCOUNTS RECEIVABLE FOR AMALGAMATED STORES INC. AT THE END OF SECOND QUARTER?

A Description (which ends in a period) is really a description of a desired report. The following are examples of Descriptions:

DISPLAY THE JOB LOCATION FOR THE EMPLOYEES WHOSE PROFIT SHARING STATUS EQUALS ELIGIBLE.

PLOT THE TREND OF THE NIELSEN AUDIT MARKET SHARE DIVIDED BY THE TOTAL $ EXPENDITURES FOR ADVERTISING FOR ALL LOCAL MEDIA ; THE PROPORTIONATE SHARE FOR NATIONAL MEDIA FOR THE CHICAGO METRO MARKET FOR ALL QUARTERS OF THE SIXTIES FOR CORNY CORN FLAKES AND ALSO THE COMPETITIVE BRANDS.

GIVE ME THE VALUE ADDED, NUMBER OF EMPLOYEES, VALUE ADDED DIVIDED BY TOTAL SALARY AND ALSO CAPITAL TURNOVER FOR THE DIVISIONS WHOSE RATIO OF CAPITAL TURNOVER TO THE CORPORATE TOTAL IS GREATER THAN 1.0 AND WHOSE PROFIT MARGIN IS GREATER THAN 15.5. TITLE THIS REPORT "PRODUCTIVITY ANALYSIS".

In the case of Descriptions, the response will not be given until the report is actually asked for.

The preceding examples illustrate the simple English use of MUSE.

Also like English, MUSE provides for great flexibility in synonym and phase structure. It allows for differences of vocabulary according to the preference of the user, abbreviations for familiar words or for phrases that take too long to spell out, and the executive's usage of superfluous words or phrases.

The user of MUSE need not worry about mistakes in construction. MUSE is its own editor, and a "forgiving" one at that; it discovers mistakes and immediately prints them out for correction. Dialogues can be easily modified. MUSE has the capacity of going from dialogue to report, back to modify dialogue, and then back to report—all within moments. In this interaction, there is a much greater time saving than just the "turn-around" time would indicate. The psychologically unattractive and time-consuming period for rethinking the problem is no longer necessary.

Dialogues may turn out to be so simple that there is no particular point in preserving them for general use (e.g., "WHAT ARE THE SALES FOR THE XYZ DIVISION FOR JULY OF 1969?"). On the other hand, an executive may create a dialogue that is of great continuous and general use to him and others in the firm. In such instances, the dialogues may be preserved so that a decision process becomes a tangible asset of the firm. Moreover, in MUSE even complex dialogues can be improved upon without the danger of "locking in" to the highly structured and inflexible format that characterizes so many applications programs available today.

QUESTION How does the report-generator in MUSE operate?

ANSWER Just as the language of MUSE is designed for the ease of the user, so the report-generator is designed to relieve the user of problems of formating. The formating of reports is built into the system; the user does not have to be concerned with page layout, field editing or identification, or spacing. The system takes the responsibility for producing attractive and readable reports. Further, the system can offer many formating options, so that the customer may specify a particular layout or a particular kind of spacing or a particular unit calculation if he so desires.

MUSE is uniquely comprehensive in keeping track of the units and scaling of all data as it passes through the system and onto a report.

> *Units* are the measurement in data such as feet, meters, tons, hogsheads, liters, dollars, kilowatt hours, francs, M. P. H., etc.

> *Scaling* is the word representation of the number of zeros that have been dropped off the end of a figure, e.g., $6 million instead of $6,000,000.

Units and scaling factors are captured when the data are loaded and, with every calculation, parallel resolution is made to the units and scaling of that answer. It is then possible, for instance, to combine French Francs and Dollars, and have the answer expressed in either currency without user conversion. That answer will also be identified as to decimal scaling, which is a clear advantage. In other systems it would be left to the user to decide whether the answer is in thousands, millions, or whatever.

Should basic unit rules be violated (such as adding M.P.H. to liters) the user will be protected by a warning from the system.

Unlike other systems, then, MUSE enables the user to disregard units and scaling calculations. Since the operation is completely automatic, there need be no concern or conscious effort on the part of the user.

QUESTION How is a corporate data base established?

ANSWER All the corporate data that can have impact on top-level corporate decisions are collected, maintained and manipulated by MUSE through an organized procedure. Data can be either new or "updated," so that the data base can change with changing corporate demands. Any part of the data base that has become obsolete or irrelevant can be flushed out of the system. Executives within the same firm can have simultaneous access to whatever corporate data they need, with, of course, built-in safeguards to prevent information retrieval by unauthorized personnel. Data retrieval can be restricted along functional lines; management can do whatever it wishes to control access to corporate information.

Any data that exist in a machine readable format can be loaded into the computer in a short time by conversational interaction similar to that occurring between the executive and the computer. The person loading the data is *not* the executive, who need not concern himself with the data-loading procedure. Instead, the person trained to do the loading need only be familiar with the input format and the names to be assigned to the data. Data can be updated through the terminal by anyone having access.

QUESTION In what way is MUSE a "proprietary" product?

ANSWER The MUSE system is proprietary in that it is owned by Meta-Language Products, Inc., and can be obtained on a licensing arrangement by individual corporations and agencies. Rental or lease arrangements can also be made.

The fact that MUSE is a system than can be obtained *immediately* eliminates the high expense of an individual corporation's developing such a system. It is estimated that MUSE will cost only about 1/10 or less than what such a system would cost if a corporation could allocate the programing talent and hardware necessary to accomplish such a monumental task. MUSE obviates the necessity for research and development at great expense over a lengthy period of time, as well as the cost of updating and training. Also, there are no continuing software costs in MUSE, since its use constantly develops applications programs.

QUESTION Who are the persons who developed MUSE?

ANSWER George W. Potts, David K. Homrighausen and Peter Kaiser are the principal members of the systems development team at Meta-Language Products, Inc.

MUSE was conceived chiefly by George W. Potts, President and Chairman of Meta-Language. Mr. Potts received his A.B. degree in mathematics from Dartmouth College in 1960. He later attended the New York University Graduate School of Business Administration. As an officer of the Bank of New York, he was responsible for the application of computers to investment analysis, economic research and bank planning. In 1965, at White, Weld & Company, he co-authored with David Homrighausen and two others the "First Financial Language" and had many primary responsibilities in establishing the "Analystics" group, now called Interactive Data Corp.

David K. Homrighausen, Vice President, Systems, received his A.B. in philosophy from Cornell University in 1956, attended Columbia University's Graduate Faculties and the New York University Graduate School of Business Administration. In 1962, at William S. Morris & Company, underwriters, he developed and implemented models for the computer evaluation of municipal and corporate bonds. Later, at White, Weld & Company, he worked with George Potts on the "First Financial Language," and subsequently worked as a senior systems analyst at Market Monitor Data, Inc.

Peter Kaiser, Senior Systems Analyst, attended the University of Utah, Princeton University and the University of California at Berkeley. He has been a programer at Kennecott Copper, Princeton University and the University of California. Most recently he has been a systems programer and lecturer at Columbia University's Computer Center and has taught a computer language and applications in a graduate course there.

Other officers of Meta-Language are Harold C. Kurfehs, Vice President and Director of Marketing, and Fairfield W. Hoban, Vice President, Secretary and Counsel.

Mr. Kurfehs holds a B.S. degree from St. Peter's College and an M.B.A. from the Wharton School at the University of Pennsylvania. He spent ten years in marketing-advertising at The American Tobacco Company as a Product Manager; at Benton & Bowles, Inc., as an Account Executive, and most recently at Wells, Rich, Greene, Inc., where he was an Account Supervisor. He has participated in the strategy formulation, planning and execution of marketing-advertising programs for seven of the nation's largest corporations.

Mr. Hoban is a graduate of Columbia College '44 and Harvard Law School '48. After practicing law in New York State for several years, he served as chief counsel for the New York State Employee Welfare Fund Agency and as Deputy Director of the Peace Corps in Pakistan. He has represented several firms in the television and investment banking fields.

Members of the Board of Directors of Meta-Language include:

Walter Taradash, Princeton '36, general partner in charge of corporate financing, mergers and acquisitions for Ingalls & Snyder, a member of the New York Stock Exchange. For many years Mr. Taradash served as financial counsel to The Diebold Group, Inc., a prominent computer consulting firm.

Peter Campton, CCNY '51 and Harvard Law School '54, assistant manager of the legal department of Merrill Lynch, Pierce, Fenner and Smith, Inc.

Alfred Clark, Boston University (BA) '66, Columbia University School of International Affairs (MA) '68, Columbia University School of Business Administration (MBA) '69. Mr. Clark is with Loeb, Rhoades & Co. and is a director of Interactive Sciences Corporation and the Clark Estates.

GUIDE TO COMMONLY USED COMPUTER TERMS

On-line refers to the use of a remote terminal—a teletype or television-like device—installed in the office or even the home of a user which is connected by an ordinary telephone line to a computer. Current dialogues or data can then be introduced directly into the data processing system as they originate.

Real time is the span of time required for the information from the computer to be of value. It is the amount of time required in real life situations, *not* the time required by the computer. The time-sharing aspect of MUSE, with its immediate information update and retrieval, provides the customer with "real-time" capability.

Time-sharing means the simultaneous access to a computer by multiple users. Each user has the impression of being in communication with his own computer; this is what is meant by "interactive environment." The executive who has a remote terminal converses with the computer by typing—or having his secretary type—his MUSE Commands, Questions and Descriptions onto the terminal keyboard; the computer responds on the same terminal. Since the time of a high-speed computer is being shared by many users, no user is spending the entire computer capability for his own particular problem.

The idea of using central computers to service remote terminals originated in the '40s, but time-sharing has really come into its own only recently, due to increased difficulties of pushing non-routine jobs through the EDP department, using batch-processing machines.

In *batch-processing*, each job, while complex and requiring skilled personnel, is routinely and repetitively processed from start to finish. Batch-processing operates something like a laundromat, to which a customer brings batches of calculating and computing work and then either waits or comes back for the solution. "Turnaround" time from submitting a problem to delivering the answer can take hours, days or weeks.

Conversely, in the time-sharing environment answers are immediate.

The use of MUSE, however, in no way competes with the usual routine EDP procedures of a user-corporation. In fact, it complements the routine procedures in that it is concerned with and applicable to *non*-routine problems and decisions for which immediate information is necessary.

Designed by Sellsbury Associates. Photographs by Fred Burrell, Ewing Galloway and Charles Phelps Cushing.

EXECUTIVE SUMMARY

MUSE

MUSE

TECHNICAL SUMMARY

MUSE* TECHNICAL SUMMARY

(Version 1.0 October 15, 1969)

The format of this technical description of MUSE is based on the one outlined in the Feature List (Part 2) of "A Survey of Generalized Data Base Management Systems," a Codasyl Systems Committee Technical Report published May, 1969 and available from The Association for Computing Machinery, 1133 Avenue of the Americas, New York, New York 10036. (As stated therein, "The price is $7.00 and all orders must be prepaid.") Reprints of only the Feature List are available from Meta-Language Products, Inc., free of charge, to those who need further background and enumeration of the format procedure. This is not meant to be an endorsement of MUSE by the Codasyl Systems Committee nor, in fact has MUSE been submitted for evaluation by this committee. Rather, this format was chosen to provide easy comparison with other Generalized Data Base Management Systems.

*MUSE is a registered property of Meta-Language Products, Inc. Copyright © 1969 by Meta-Language Products, Inc. All rights reserved. No part of this publication may be reproduced by any means, nor transmitted, nor translated into a computer language without the written permission of Meta-Language Products, Inc.

1. INTRODUCTION TO THE SYSTEM

1.1 *Identification*

MUSE (*Machine-User Symbiotic Environment*) is a proprietary software product developed and owned by Meta-Language Products, Inc.

1.2 *Status*

MUSE is currently partially operational on the Tymshare, Inc. X.D.S. 940 time-sharing computer located in Englewood Cliffs, New Jersey. It will also be implemented on computers from among the following: X.D.S. Sigma 7; Control Data 3300; I.B.M. 360/50-67; D.E.C. PDP-10; R.C.A. SPECTRA 70/46; UNIVAC 1108; and G.E. 635; not necessarily in that order. The development period for MUSE will last until February 1, 1971 during which time both across-machine-line capability and specifications improvements will be accomplished.

1.3 *System background*

MUSE has been under development since March, 1968. Its rate of progress and scope of development have been expanded twice during this period: first, in August 1968 when Tymshare, Inc. and Meta-Language Products, Inc. reached an agreement of mutual co-operation, and second, in February 1969 when Meta-Language Products, Inc. received private financing through Ingalls & Snyder, investment bankers. Half of the systems development team working on MUSE were a fundamental and directing part of the White Weld effort in the natural language and data base service called Analystics (including FFL—First Financial Language).

1.4 *Major characteristics*

1.4.1 *Data structure class*

The data structure of MUSE is based on a tree structure of partial identifiers used to map the entire data library.

1.4.2 *Generalized process provided*

The MUSE system contains the MUSE language for general interaction (questions, commands and declaratives) a data loading/updating procedure; and a report generator (including graphic capability).

1.4.3 *Language type*

The MUSE language is non-procedural and uses natural English words and syntax. The data loading/updating interaction is a series of questions by the system and answers by the user to capture data format and attributes.

1.4.4 *Language forms*

The MUSE language is free-form, comprised of English sentence types: declarative, imperative and interrogative ending in "." , "!" and "?" respectively. These sentences can be given to the system individually or in paragraph form. The sentences are, like English, formed of subjects and predicates, clauses and phrases; nouns, verbs, adjectives, pronouns, adverbs, prepositions, conjunctions, etc.

1.4.5 *Storage structure class*

The data are organized into a purely random file (better called library) of a size determined by a two-word addressing scheme (on the X.D.S. 940 this places a theoretical upper limit of 2^{48}, but a practical limit of about 64 million data elements). Referencing indices are detached from the data and placed on faster storage. The actual structure of the data is never apparent to the user.

1.4.6 *Modes of use*

The MUSE system is designed to reside in a general purpose time-shared environment. The user describes and generates reports on-line or to other attached devices. Data updating can take place in real-time from remote terminals. Data loading can take place in time-sharing but it is advised to load data during off hours. If batch processing is provided by the T.S. monitor of record then MUSE can behave as much like batch processing as is allowed.

2. DATA STRUCTURES

2.1 *Items*

2.1.1 *Systems term for items*

The MUSE term for item is "DATA ELEMENT".

2.1.2 *Item naming*

A data element in the MUSE system is an atomic member of a N-dimensional array (O<N<16) whose coordinates are called "IDENTIFIERS." An identifier is either a word or a series of words not exceeding five words. Words are delimited by blanks and may contain all alphanumeric characters and seven special characters (# $ % & . ' @). Entirely alphabetic or entirely numeric words may contain ten characters; others are limited to eight. Examples of identifiers are GENERAL MOTORS INC., TOMATO JUICE, and 4TH QUARTER. It is natural to expect the user to be aware of the implicit dimensional structure of data in naming data elements, but the user is not required to be aware of files or other data structures. In naming data elements, identifiers are concatenated by means of qualifying operators. In the MUSE system, qualifying operators are borrowed from ordinary English usage. Two such operators are FOR and 'S. For example, a data element might be called GNP FOR ENGLAND or ENGLAND'S GNP. At times, the use of English-language qualifying operators may suggest a logical disparity between identifiers. The system, however, does not always require good English. XEROX FOR SALES may be acceptable. For similar reasons, identifiers should be nouns or noun phrases. Any identifier can receive a synonym or abbreviation at *any time* (not just at data loading) from a remote terminal.

2.1.3 *Item data types*

Data elements are textual or numeric. Textual elements are *only* single words (such as BLUE, FEMALE, GOOD . . . see 2.1.2). Numeric elements may be floating-point or integer. Integers may be cardinal or ordinal. Dates (6\6\62) and times (4:06:53) are special cases of ordinals.

2.1.4 *Data variability*

Virtually any numeric data can be loaded into the MUSE system (number bases from 2 to 16; various encodings--ASCII, BCD, HOLLERITH, EBCDIC, etc.) but only 10+ significant decimal digits are maintained. Textual information is limited as mentioned in section 2.1.3.

2.1.5 *Multiple-valued items*

A single identifier before qualification may refer to a multitude of data elements, a specific element being chosen through either default or further qualification in the dialog (i.e., PRODUCTION FOR 1965 as opposed to PRODUCTION FOR 1964). However, a specific level of the qualification process can only reference one data element.

2.1.6 *Sub-items*

There are no sub-items in MUSE except the components of dates and times.

2.2 *Groups*

In MUSE, a group is a list constructed by the use of data element identifiers and/or previously constructed groups. These groups have identifiers in the same manner as data elements.

2.2.1 *Systems name for group*

In the following paragraphs groups and group identifiers will be called "GROUPS" or "LISTS." A list of all identifiers in a given logical dimension is frequently referred to as a "CLASS."

2.2.2 *Group structure*

Groups have no limitation as to structure. It is up to the user to form meaningful and non-recursive groups. The concept of "repeating group" is not helpful to the user except insofar as it can be used to translate physical dimensionality to logical

dimensionality during data loading. A natural distinction occurs in the MUSE system between (slow access, large) and (fast access, small) groups either by using a dialog as a list expression (identified in the same manner as data elements, etc.) or through the use of the EQUATE command.

2.2.3 *Group relationships*

Groups may be nested to any level consistent with storage and timing considerations. Groups need not relate to a single data structure (as loaded).

2.2.4 *Group identification*

Within a class there could be an identifier for a data element which further identifies one of the other classes in the logical array. This is *not* necessary for proper use of MUSE and will occur only when these data elements (which frequently appear in physical arrays of data for referencing checkout) are retained inadvertently through a data loading.

2.2.5 *Types of groups*

The types of groups are outlined in 2.2.1. An example of a list would be TOOLS being equated to HOE, RAKE, SHOVEL, AX. An example of a class would be the universal set of YEARS.

2.3 *Entries*

Entries have no meaning in the MUSE system. The CODASYL format for describing data structures assumes that the concept of a physical file is retained in the data management system described. It has already been shown that to MUSE the *logical* structure of the data is paramount. Its *physical* structure is important only insofar as it must be described in the data loading interaction of MUSE. The rest of the outline entries under 2.0 will be used to explain features of the MUSE system that frequently correspond only tenuously to the CODASYL format.

2.3.1 *Systems term for entry*

In the MUSE system one should probably call the repeating group of the logical data array with the largest periodicity in the input structure an "ENTRY". However, since a logical data array frequently is unconsciously rotated by the data base designer to a physically convenient arrangement, an entry is better defined as any coordinate point along any dimension of the logical data array. Since we have already made the distinction between the data element of a concatenation of identifiers (or entry points) and these identifiers themselves, there is no need for clinging to this term.

2.3.2 *Entry types*

If one speaks only of physical data files as they are entering the MUSE system then multiple entry types are acceptable if they are consistent across a repeating group (or logical coordinate group). Entry types are distinguished as to form when the data is loaded, and then no longer needed. These can be *fixed* length fields, records, segments or files; *variable* length fields, records, segments or files, delimited by a character count, field count, record count or segment count entry *OR* by a termination character; or mixtures of both.

2.3.3 *Entry identification*

See 2.2.4, as entry identification is exactly equivalent. Note, however, that an entry (or group) identification that is imbedded in the data structure can be used in lieu of a name supplied at the terminal at data load time. However this is a risky procedure, as the person responsible for data loading gives the system carte blanche to update the dictionary without visual review.

2.4 *Files*

2.4.1 *Systems term for files*

The term "file" is superfluous in the MUSE system except to refer to a contained data structure at data load or update time.

2.4.2 *File types*

As new data elements correspond precisely, identifier for identifier, with a data element already in the data library then that data element will be "updated" with the loading of new data values. Non-symmetric matching of identifiers causes new data array elements to be created.

2.4.3 *File identification*

Files lose their identification once they enter the MUSE data library except as they exist in the dialog that was used to load them. This dialog can be used to load identical data, update this data, or delete the entire data array from the library. Data, of course, can be removed with other mechanisms.

2.5 *Data structure generalization*

Although asked not to mention it here in the CODASYL Feature List it is interesting to note that data elements that are alphanumeric could theoretically be used as data referencing mechanisms 1 to N levels removed as in indirect addressing for machine language operands. Another general data structure capability is, of course, the list, arithmetic, boolean and other expression capabilities (with identifiers attached) which were previously touched on. These then are equivalent to secondary data elements but, in fact, are procedures in conjunction with primary data or other expressions.

2.6 *Data security*

In MUSE, security is defined at the file level at data load time. Functional restriction is permitted by defining the user numbers that can reference a particular data array. All previous restrictions are overriden for duplicate data (i.e., updated portions).

2.7 *Other data structure features*

One particular type of identifier may be operated on arithmetically to effectively change the data referred to. This is an *ordinal* integer (which among other things can represent a date or time) to which *cardinal* integers may be added or subtracted (i.e., 4\57 + 7 MONTHS). Note that ordinal numbers may be used as identifiers and therefore can have only very simple unit attributes.

3. FUNCTIONS

3.1 *Language form*

The teletype character set is the base of the MUSE language. It is broken into four major categories:

1. Alphanumeric A-Z, 0-9 and the special characters # $ % & . ' @. The subsets of pure alpha or pure numeric have certain advantages (see 2.1.2)
2. Operators (including interior punctuation) " ( ) • + , – / : ; < = > ↑ \
3. Sentence termination punctuation . ! ?
4. Special line editing echos [ ] ← X

Spaces may (but need not) indicate the delimitation of language tokens (i.e., SALES FOR FORD as opposed to SALES+EARNINGS). Two spaces are used to separate sentences in the standard typographical format. The syntax of questions, commands and declaratives is ordinary English (e.g., for a command it is <COMMAND VERB> [<PREPOSITION>] [<ADJECTIVE> | <ADJECTIVAL PHRASE>] [<OBJECT> | <OBJECT PHRASE>] [<PREPOSITION>] [<ADJECTIVE> | <ADJECTIVAL PHRASE>] [<OBJECT> | <OBJECT PHRASE>] . . . [<PREPOSITION>] [<ADJECTIVE> | <ADJECTIVAL PHRASE>] [<OBJECT> | <OBJECT PHRASE>]!). Control is returned to the system (for syntax checking, etc.) when a carriage return is typed immediately following a termination punctuation. If the MUSE system has asked the user a direct question then carriage return alone is sufficient to return control. Text may be continued on following lines by merely typing carriage return *or* line feed. If the user forgets this the system will automatically supply carriage return, line feed before characters are overtyped.

The user has *full* line editing capability at *all* levels of a MUSE interaction. This allows him to correct mistakes in typing or fix previous text by using simple control character conventions very similar to those in the QED system developed at the U.C. at Berkeley under Project GENIE. MUSE always indicates the user's turn to type with a left justified "?".

3.2 *Data structure definition*

The structure of the data as it enters the MUSE system is described in the data load dialog. Here the system asks for the input device, the physical structure of the data layout, the nature of the repeating groups and their nesting (or logical structure) and how it relates to the physical structure of the data, the formats for data fields, the units and scaling, identifier information, default conditions, the form taken by "not available" information, etc.

3.2.1 *Definition of data items*

Since the item identifier frequently (but not always) refers to the field level (or repeating group with the greatest periodicity), the MUSE system specifically asks what repeating group is to assume this role.

3.2.1.1 *Definition of data item types*

Item type information is captured in the data format specification.

3.2.1.2 *Definition of data item length limitations*

Any data element which exceeds the internal storage capability of the system will be treated in the following fashion:

Alpha—right truncation to 10 characters

Alphanumeric—right truncation to 8 characters

Numbers (floating point and integer)—the 10 most significant digits

Dates and times—all proper dates and times can be handled

3.2.1.3 *Definition of multiple-valued items*

There is a special format specification for dates and times.

3.2.2 *Definition of groups*

The data loading dialog of MUSE differentiates repeating groups (or classes) with questions on logical structure. Groups (or lists) are created with the user expression capability (i.e., the command, EQUATE "TOOLS" WITH "HOE, RAKE, SHOVEL, AX"!) or with dialogs themselves that can be referenced with standard identifiers just like data.

3.2.3 *Definition of entries or entry types*

Two or more distinct *logical* structures within one physical file (which is highly unusual) will require multiple loading of this physical file, one for each logical structure. This is different from variability within a logical or physical structure (most aberrations of which MUSE can handle).

3.2.4 *Definition of file and file types*

The notion of "file" is lost after data loading with MUSE. The only handle on the phantom file is by means of the data loading dialog itself if the user chooses to save it.

3.2.5 *Definition of more general data structures*

User expressions can be used to describe simple or complex relationships or processes within the data. These expressions can be arithmetic, boolean, relational, set, descriptive (concatenations of identifiers), lists (or set-determining), functional, idiomatic, or combinations and superstructures of the above. Lengthy expressions are stored and referenced as dialogs; shorter ones are created via the EQUATE command and stored in faster storage (disk vs. drum).

3.2.6 *Definition of security*

Security is defined at the functional level by unlocking data at load time to a particular subset of all user numbers.

3.2.7 *Data validation*

As data is being physically loaded from an external file to the data library there are levels of data validation available to the person performing the loading function, each of which can be armed or disarmed at his discretion. They are:

1. Notice of any truncation, rounding or other default conditions.
2. Notice and opportunity for correction of any data that does not meet its format specification.
3. Notice and opportunity for correction of any data that falls outside of a specified range (not available data would be a special case of this).
4. Notice and opportunity for correction of any data that does not cross-foot check within the input record as defined.

3.2.8 *Revision of data definitions*

After loading a physical data structure no modification of its *internal* structure can be made. However, the identifiers that are used to reference those data, their definitions, or data-creating expressions can be easily modified. If the data loading dialog has been saved, it can then be used again to load or update similar data structures (the data loading dialog is modifiable).

3.2.9 *Other data definition specifications*

The data loader is asked to specify a rank number from 0-9 for particular, unique loadings of data. When the data library space is used up, the MUSE administrator can give the system discretion for purging data to make room for new data on the basis of these weights. At present it is not anticipated that an archival storage capability is necessary for the MUSE system, although serious thought is being given to providing a collector of systems statistics (such as the number of references for data items or arrays).

3.3 *Storage structure definition*

The user has no control of the internal structure of the MUSE data library.

3.4 *Interrogations*

Interrogation takes place in the MUSE language in the form of questions or the description/production sequence. Questions can interrogate the data base, retrieve definitions for all words in the system (including identifiers), and answer HOW and WHY queries. The description/production sequence takes the form of dialogs that, whenever the user requests it, are used to produce reports or graphics. Dialogs consist of only declarative sentences which describe the function operations, arithmetic sequences, logical combinations, relational criteria, set structures and Venn combinations, descriptive sequences (or identifier combinations), sorting criteria, title specifications, grouping and summarization needed for display of data. Dialogs are stored, retrieved, printed and modified via the MUSE command capability.

3.4.1 *Selection criteria*

In MUSE there are two distinct types of selection:

a. Data is qualified by the use of qualifying phrases (e.g. SALES *FOR XEROX*) constructed of a qualifying operator and an identifier.

b. Lists (or, in many cases, universal sets) of identifiers are screened to produce subsets for final qualification as in "a." above. This involves set expressions or conditional clauses about the eligibility process for identifiers to stay in the list. The syntax of a conditional clause is as follows:

<IDENTIFIER> IF [<RELATIONAL STATEMENT>]
[<BOOLEAN OPERATOR> <RELATIONAL STATEMENT>
[<BOOLEAN OPERATOR> <RELATIONAL STATEMENT>...]]

3.4.1.1 *Atomic condition*

In screening a list of identifiers the conditional clause, as above, is comprised of atomic conditions that include everything necessary to reference data except one class of identifiers (or list) which stands apart. This is the set being operated on as in this example: DISPLAY THE EMPLOYEES IF THEIR EARNINGS ARE GREATER THAN $12000 AND THEIR HEIGHTS ARE LESS THAN 6 FEET.

3.4.1.1.1 *Comparative conditions on item values*

The relational statement itself can contain arithmetic expressions, boolean statements, qualifying phrases or even other relational expressions! The atomic elements on either side (but at least one side) of the relational operator can be computed from data reference(s). They can also be constants imbedded in the dialog. The relational operators are:

= equal to
> greater than
< less than
>= greater than or equal to
<= less than or equal to

or their synonyms. (Please note that NOT is a general negation operator in MUSE).

3.4.1.1.2 *Existence conditions*

Any conditional statement will automatically exclude "not available" and other default data conditions.

3.4.1.1.3 *Other atomic conditions*

MUSE has the set operators UNION and INTERSECT which supply logical opera-

tions (i.e., Venn) on sets (or lists). These most often will be used in synonym form as follows:

SOLDIERS AND ALSO SAILORS	(UNION)
SOLDIERS WHO ARE AMERICANS	(INTERSECT)
OTHER THAN AMERICANS	(NEGATION)

3.4.1.2 *Item conditions*

Relational expressions can be connected with the boolean operators AND and OR to create complex conditions for list extraction (as in 3.4.1). Nesting of arithmetic, boolean, qualification, set and relational expressions can be taken to enough levels to permit any reasonable construction. Identifiers and qualification need not be repeated within complex expressions if the expression so formed is not logically inconsistent with the syntax rules for nesting. Note, however, that these conditionals can refer to many items simultaneously and, in fact, need not reference data elements that are to be included in report output.

3.4.1.3 *Entry selection criterion*

Any further statement of selection capability here would be redundant (see 2.3).

3.4.1.4 *Weighting of conditions*

Weighting of atomic conditions can be accomplished in MUSE with arithmetic expressions acting on truth values or data elements.

3.4.1.5 *Other features of selection criteria*

Any complex conditional clause can be equated to an identifier (or created as a separate dialog) and later recalled simply by using this identifier.

3.4.2 *Data extraction*

The conditional capability described in 3.4.1 relates only to an operation on a list (or class) of identifiers. It is when this list of identifiers is further qualified with other identifiers (which can be processed lists themselves) and further manipulated via the arithmetic, functional or sorting capability (as the user desires) that a report is fully defined and producible. At this point in the use of MUSE there is no indication of what file data may have been loaded from, so therefore no restriction is imposed.

3.4.2.1 *Generalized extraction features*

The general data extraction feature of MUSE is based on the use of identifiers in the qualification process to describe unique data elements to be extracted from the library and reflected in the final report. This economy of statement is analogous to the economy of using headers to describe the data within the body of a report. Each element of a report does not receive full identification along with the number itself. Rather, it is by the implicit intersection (or qualification) process with the header (column, row, page, etc.) information that the report reader determines what a particular number represents. The identifiers that may be used for this process are those present in the dictionary or those supplied at report call time.

3.4.2.1.1 *Sorting*

MUSE provides sorting capability on any list of identifiers (there is also a RANK function). As a list can be comprised of identifiers referencing other lists, then as many levels of sort nesting (or collating sequences) as the user desires can be accomplished. The user gives the SORT verb followed by the name of the list (identifier) to be arranged, then the rest of the qualifying phrase (or more complex structure) i.e., SORT TOOLS USING LENGTH/WIDTH. Descending sorts are standard by default; ascending sorts must be so stated with this adverb. The user may give any number of sort statements and if they are ambiguous the latter ones will override. Alphabetic sorts are also accomplishable on lists according to standard conventions.

3.4.2.1.2 *Item properties which are extractable*

All identifiers in MUSE, whether original with the system or user-produced, have definitions which can be recalled at will or are available in a system-generated dictionary. This gives a structure to the language which is consistent (data elements cannot be given a duplicate partial identifier which does not correspond in definition) and unambiguous.

The data elements themselves have four properties that are captured at data load, adjusted at each level of calculation and conspicuously displayed on the report output:

1. UNITS—the measurement base of the number represented such as FEET, DEGREES, K.W.H., POUNDS STERLING, etc.
2. SCALING—the word representation of the number of decimal digits shifted, e.g. HUNDREDS, BILLIONS, THOUSANDTH, etc.
3. INTEGER ATTRIBUTE—an attempt is made to preserve an integer attribute of a number if it was originally loaded as such. Division of two integers, addition of a floating point and an integer, etc., will destroy this property.
4. SIGNIFICANCE—an assumption is made that any numeric data loaded is significant to the next to last digit. Throughout arithmetic manipulation, calculations are made as to loss of significance and displayed on report output through zero rounding, changes in scaling and otherwise.

3.4.2.1.3 *Discrete extraction sets per selection criterion*

The user in a MUSE dialog can leave a token or tokens as variables (MUSE makes notes to the user if an identifier, operator, etc., is not recognized in the dictionary). When this dialog is used to produce a report these variables are requested for resolution. This gives multiple report capability from a single dialog. This facility is quite well suited for creating simulation models. MUSE dialogs are also easily modified through the command structure and line editing facilities in the system.

3.4.2.2 *Report capability*

Data may be extracted from the entire data library (given clearance) for a single report. A report has a dimensional characteristic like a logical data array. The mapping of this N-dimensional array onto essentially a two-dimensional medium (a length of paper) requires certain conventions for mapping . . . ROWS, COLUMNS, PAGES, VOLUMES, WORKS, LIBRARIES, etc., act as coordinates. Each of these coordinates receives header information automatically without user intervention. The cutting and pasting of the logical output array is dependent upon the size characteristics of the output receptor (such as a CRT, teletype or printer) and is accomplished entirely by the system. The user has the option to override this organization or the spacing conventions at report output time if he so desires. Units and scaling information are usually placed under the column headers if at all possible. The system also places at the top of each external page of report output the following information:

—Title (if any)
—Dialog name
—Who created the dialog
—When the dialog was last modified
—When the report was produced
—What physical page this is
—How many physical pages there are
—Who requested the report

A small sample to illustrate the report capability would use the dialog DISPLAY THE CLOSING MARKET PRICE AND ALSO EARNINGS FOR INFORMATRICS, SYSTEM DEVELOPERS AND ALSO ISM FOR 1968.
To produce the report:

1968	CLOSING MARKET PRICE ($)	EARNINGS ($—THOUSANDS)
INFORMATRICS	16.50	−23.5
SYSTEM DEVELOPERS	NOT AVAILABLE	−35.6
ISM	5.75	− 1.3

3.4.2.2.1 *Content line*

Content lines are specified as in the above example. The implicit ordering that the user places in his dialog statement is maintained unless otherwise overridden with a sort, set or conditional statement.

3.4.2.2.2 *Titles*

Titles can be included in a report with a statement such as TITLE THIS REPORT "EARNINGS COMPARISON". These titles will be centered.

3.4.2.2.3 *Heading lines and footing lines*

Heading lines are supplied unconsciously by the user when he describes the output desired. Footing lines are supplied if the user requests totals or averages as feet to columns or rows, etc. No pure footnote facility is planned.

3.4.2.2.4 *Other user specified text*

As data elements can be textual, so can they be used as report elements. However, it is plain that one should not try to perform calculations with text. Other textual notations appear when certain error conditions occur in the data manipulation process. They are:

NOT AVAILABLE
UNCALCULABLE
TOO BIG
NOT SIGNIFICANT
OVERDEFINED
CLASSIFIED

3.4.2.2.5 *Editing and formatting*

The purpose of MUSE's assuming a dominant role in formatting report output is that it has more information at its disposal for maximum information content in a particular output. The user has the option of changing the units and/or scaling of a particular coordinate, rotating the presentation arrangement along any axis or expanding or contracting spacing. Of course, by altering the dialog he may change the ordering along any coordinate.

3.4.2.2.6 *Derived data*

The MUSE system has a complete set of point and vector functions for familiar computations. They are:

POINT FUNCTIONS — LOG, ANTILOG, FRACTIONAL PART, REAL PART, ABSOLUTE VALUE, SINE, COSINE, TANGENT, COTANGENT, SECANT, COSECANT, INVERSE, SQUARE ROOT, FACTORIAL

VECTOR FUNCTIONS — MAXIMUM VALUE, MINIMUM VALUE, AVERAGE, MODE, MEDIAN, TOTAL, NUMBER, VARIANCE, STANDARD DEVIATION, KURTOSIS, RANK

3.4.2.2.7 *Other report facilities*

MUSE will have graphic display capabilities by 1970. This will enable users to produce histograms, plots, scatter diagrams, bar charts, pie charts and other useful analogs of numeric output.

3.4.2.3 *Extraction of files for use outside the system*

Normal formatted output will go to CRTs, teletypes or printers. However, the user has an option to output a report in unformatted form to a file medium (disk, cards or tape). This means that the array of only numeric information (no titles or headers) is arranged in sequential fashion with notations to the user for locating individual items. These files can be referenced by other programming languages (BASIC, FORTRAN, etc.) or special applications of these languages such as linear programming models, regression analysis, etc.

3.4.2.4 *Extraction of files for systems use*

There is no need for this facility as the user can have reference to all the data at once.

3.4.2.5 *System triggered outputs*

There is no facility in the original version of MUSE for system triggered outputs. However this area will be given serious consideration for future implementation. (Note that this suggests a system dial-up capability.)

3.4.3 *Invocation of predefined interrogations*

All dialogs that produce reports are automatically preserved if the user so desires. All he need say is RECORD THIS DIALOG! and assign it an identifier for referencing. At any time he may refer to this dialog to produce reports or to alter the text itself (even in the middle of other dialogs!). He might say CHANGE THE DIALOG WITH THE NAME OF "BALANCE SHEET"! Also note the data loading dialogs are automatically preserved and are self-modifiable but in a question-and-answer fashion.

3.4.4 *Other features of the interrogation process*

As mentioned earlier MUSE allows the user to create and reference system global expressions that can be used to describe secondary data procedures (arithmetic calculations, user-created unit constructions, user constants, lists of identifiers, 1 for N synonyms, relational constructions or other more complex structures) using any legitimate operator combination in the system. When the system is installed there is a small set of system expressions available immediately which refer to units and scaling constructions and certain universal constants. All these expressions are referenced simply by their identifiers.

3.5 *Update*

Data may be updated in the MUSE system in real time using a command like CHANGE THE VALUE OF "CONSUMPTION FOR PEANUT BUTTER BY TEENAGERS" TO "15.5 TONS"! Gross data update may also be accomplished by the same dialog that was used to load the original data base (in either original or modified forms). If a new data base is being entered and it has single data elements that exactly correspond to existing elements then these are considered as updates to the data library (the person loading the data can disarm this mechanism).

3.5.1 *Selection criteria*

Selection criteria for the interactive updating of data may only take the form of an identifier with qualifying phrase(s).

3.5.2 *Updating specifications*

Assuming clearance the user may update, delete or inspect individual data elements from his terminal. The variety of ways is discussed in 3.5.

3.5.2.1 *Entry level*

This is not relevant except as to the discussion of entries (see 3.2.3) and their updating using old data-loading dialogs.

3.5.2.2 *Group level*

The updating of items is discussed in 3.5.

3.5.2.3 *Intra-item level*

If, in the interactive environment, the user types CHANGE THE VALUE OF "SALES FOR HULA HOOPS"! the system will type out the data element corresponding to the quoted portion (if valid) and await the user's correction. He then has editing control and may correct a single internal character by hitting just a few keys.

3.5.3 *System triggered updates*

No data may be updated automatically but data may be deleted automatically (see 3.2.9).

3.5.4 *Update validation*

All gross data updates are validated in exactly the same fashion as new data loadings. Interactive updates are validated by the user visually.

3.5.5 *Invocation of predefined updates*

The data-loading dialog may be preserved in the same fashion as report-generating dialogs. These can be used over and over assuming the same or similar form to the input file.

3.5.6 *Audit trail*

The audit trail facility of MUSE may be armed when new data is being loaded. If any data is found to exist already in the system data library then a warning will be given to the person performing the loading. This will note the old and new values and ask for permission to change the value. The hard copy from this terminal can be used as an audit trail.

3.5.7 *Other update facilities*

The system maintains its own data library and periodically refreshes the storage and indexing structure for optimal speed and space saving. The system also allows the user to remotely change identifiers, other tokens or synonym definitions or to rename identifiers.

3.6 *File creation*

MUSE does not create any of its own files other than the global library. There is no user intervention necessary to start this process. It begins with the first data load.

3.6.1 *Format of entry level input data*

Virtually any format of input file can be accepted by the MUSE system if this file has been created for data processing. In the development of this generalized interface the designers of MUSE have encountered a few formats that MUSE could not accept, but these were designed for producing publications.

3.7 *Global functions*

Global functions capability has already been mentioned in sections 2.2.2, 3.2.2, 3.2.5 and under 3.4.1.

3.7.1 *Arithmetic computation*

The arithmetic operators in the MUSE system are:

+ addition
- subtraction
* multiplication
/ division
↑ exponentiation

They may be used within a sentence at various nesting levels and to any complexity consistent with MUSE's syntax rules (which are consistent with standard algebra textbook notation). Some examples are:

SALES–COST OF GOOD SOLD
SALES FOR WESTERN REGION/SALES FOR EASTERN REGION
LOG FOR (PRICE/EARNINGS)

Parentheses are used when the natural nesting structure of the various operator types (POINT FUNCTION, ARITHMETIC, LIST, SET, VECTOR FUNCTION, QUALIFYING, MATRIX FUNCTION, RELATIONAL, BOOLEAN, ASSIGNMENT, CONDITIONAL) must be overridden.

3.7.2 *Own code*

There is no provision for entering a procedural language from MUSE except through the unformatted file interface.

3.8 *Other functional capabilities*

Since MUSE is an interactive language it must constantly monitor the user for his mistakes. In fact the system can be used by a small child typing randomly without any likelihood of harm being done to the system or the data library. The various general levels of errors and their correction procedure follows:

–Typing errors	these can be corrected immediately via the line editing capability.
–Character string errors	if the system cannot parse (break apart) a character string into language tokens then various substrings are returned to the user for correction until all the parsing rules are satisfied.
–Syntax errors	various syntactical checks are made on the completed sentence to catch sloppy construction. Here, nesting and juxtaposition illegalities, unequal parentheticals, etc., are noted to the user. Any syntax error must be corrected then as the sentence is returned to the user for editing.
–New words	words or phrases which cannot be located in the dictionary are noted to the user and must be supplied before output can be accomplished.
–Semantic errors	sentences may have proper syntax but no meaning in the real world (e.g., CHANGE THE SENTENCE OF A WORD WITH THIS VALUE!). These are noted to the user immediately in a command or question and the user is left to restate or not restate at his discretion. These types of errors in declaratives will usually result in bad definition of data elements.
–Errors of misinformation or misapplication	underdefinition of data elements may result in the use of default data elements. Overdefinition results in the "OVERDEFINED" notation on report output. Other report notations as mentioned in 3.4.2.2.4 result from taking logs of negative numbers, referencing "not available" data, etc.
–Logical errors	the only logical lapses at a high level that MUSE will catch are with the units mechanism. Here inconsistent arithmetic modeling will frequently create garbage units.

Questions about these errors or system procedures in general can be asked in free form and the system will attempt to give meaningful answers.

MUSE questions are powerful enough to replace a desk calculator—even the programmable electronic ones. Look at these examples:

WHAT IS (5.3/292.5356) +.0793?
WHAT IS THE LOG OF 3.1415915.7?

4. STORAGE STRUCTURE

The storage structure of the MUSE data library is *never* apparent to the user. Since MUSE is a proprietary product it is necessary to be quite general in this section.

4.1 *Item level storage representation*

Data elements are stored as word pairs (in most cases double precision floating point) on the disk or other direct access medium.

4.2 *Entry and group level storage structure*

The organization of data is random. There is no indexing information stored with this data. The indexing overhead is far less than 1% and is stored on faster storage.

4.3 *File level storage structure*

There is no physical retention of a file structure evident once it has been loaded. It is likely that data from one loading might tend to be located within certain disk sectors or sector groups but this is of no consequence.

4.4 *Multiple file storage structure*

The indexing scheme for multiple referencing is confidential.

4.5 *Other storage structure features*

None.

5. OPERATIONAL ENVIRONMENT

5.1 *Hardware parameters*

The prototype MUSE system runs on the XDS-940 which in many respects is a minimal machine. Most other computers mentioned in 1.2 have more basic hardware capability than the 940 (but frequently less sophistication in the TSS itself).

5.1.1 *Minimum basic system*

The XDS-940 used to develop MUSE has:

- –64K of core
- –a real time clock
- –multiple access to memory
- –memory interleaving
- –selector channel
- –multiplexor channel

5.1.2 *Storage media*

Minimal storage capability suggested is:

- –4 million characters of drum (or other fast storage)
- –2 tape drives
- –1 card reader
- –64 million characters of direct access disk (less than 100 millisecond access rate)

5.1.3 *Terminal equipment*

Teletype (KSR or ASR) equipment or CRTs with teletype character sets can use the MUSE system either remotely, using telephones, or as hard-wired extensions of the communication interface. Certain plotting devices also have teletype interfaces and are planned as output media for MUSE.

5.1.4 *Hardware transferability*

The MUSE system has been designed almost entirely with macros and programmed operators which eases the problem of transferability from one machine to another. The XDS-940 has only 3 program-controlled registers, a small hardware instruction set, and a 24-bit word which makes it a common denominator for expansion to machines with greater capability. Work is currently proceeding on algorithms for transfer to the assembly language of other hardware.

5.2 *Operating system parameters*

MUSE makes no attempt to assume any of the functions of an operating system or time-sharing system. In fact a major objective has been to make MUSE as hardware insensitive as possible. This means that *all* I/O functions take place through an interface with the monitor and executive system. The random accessing of the data library has been so structured that systems without this feature could be circumvented and MUSE could go directly to disk. Other facilities (such as SYSPOPS on the 940) may or may not be present in other time-sharing systems that MUSE runs under. However, if they are not, they will be programmed into MUSE at that point as extensions of the interface.

5.2.1 *Basic required operating system*

MUSE will be implemented on the manufacturer-supported TSS for the computer used (in the case of IBM equipment this qualification may have to be relaxed).

5.2.2 *Significant features*

See 1.4.9.

5.2.3 *Transferability between operating systems*

MUSE in most cases will only be implemented with one time-sharing system per machine (except the 940).

5.3 *Restart and recovery capability*

MUSE will have panic button termination and recovery capability on most

computers. Also in the same fashion recovery can be accomplished after return to the executive.

5.4 *System operation reports*
There are no crystallized plans for this type of capability (see 3.2.9).

5.5 *Other environmental parameters*
None.

TECHNICAL SUMMARY

MUSE